ONLINE SYSTEM FOR CONFIGURATION AND PERFORMANCE OF MEASUREMENT REQUESTS

Field of the Invention

The field of the invention relates to physical systems and more particularly to the modeling and testing of physical systems.

Background of the Invention

In the creation of physical systems, a designer typically creates a rendition of the physical system on drawing paper. Often many drawings are required to describe the physical system.

Once the system is designed, construction of the system may begin. Where the system is small, a model maker may make and test the system to understand its operation under a number of test conditions. Where system operation differs from that expected, it is often necessary to redesign the system, build another model, and re-test that system.

Software programs are sometimes available wherein a programmer may enter simulation values and which allows certain aspects of the system to be simulated. While these programs are useful in modeling small portions of some systems they typically cannot accommodate the vagaries of larger, more complex systems.

While physical models of systems are tedious and time consuming to build, they tend to provide the best information regarding the performance and the unknowns associated with system design. Accordingly, a need exists for a means of quickly and easily assembling physical systems automatically.

Summary

A method and apparatus are provided for assembling and operating a physical system having a plurality of structural elements and structural interconnections from a remote location. The method includes the step of creating a graphical representation of the physical system at the remote location showing the elements and connections of the system to be assembled. The method further includes the steps of converting the graphical representation into an element list delineating the elements and the interconnections, transferring the element list from the remote location to an element controller and assembling and operating the system by the element controller in accordance with the element list.

Brief Description of the Drawings

FIG. 1 is a block diagram of the modeling system in accordance with an illustrated embodiment of the invention;

FIG. 2 is a block diagram of a server of FIG. 1;

FIG. 3 is a screen of a terminal of FIG. 1;

FIG. 4 is a partially completed screen of the terminal of FIG. 1;

FIG. 5 is a screen showing a circuit on a terminal of FIG. 1;

FIG. 6 is a screen showing a circuit and netlist of the system of FIG. 1;

FIG. 7 depicts a transaction flow diagram of the system of FIG. 1;

FIG. 8 depicts a circuit that may be modeled by the system of FIG. 1; and

FIG. 9 depicts test results of the modeling system of FIG. 1.

Appendix I provides information regarding the classes and methods of the server program of the system of FIG. 1.

Appendix II is source code for the main routine of the system of FIG. 1.

Detailed Description of a Preferred Embodiment

FIG. 1 depicts a block diagram of a remotely operated modeling system 10, generally in accordance with an illustrated embodiment of the invention. Under the illustrated embodiment, a user working through a terminal 12 may create a graphical representation of the physical system to be modeled or investigated. The graphical representation may include any number of physical devices interconnected to form a functional system. One or more forcing functions may be added to the modeled system to determine a response under load. A set of measurement points for evaluating system performance may be identified and added to test the modeled system. Within the terminal 12, the graphical representation of the system is converted by a conversion processor into an element list 20 and transferred by a communication processor through the Internet 14 to an element processor (i.e., a measurement resource broker (MRB) 16).

Within the MRB 16 the element list is checked for errors. The MRB 16 determines whether it has the physical elements and particular forcing function(s) needed to construct and operate the physical model. If it does, the MRB 16 assembles the physical system. Test instruments are connected to the physical model, as is the forcing function. Measurements are collected on

WHAT measures

system performance and the measurements returned to the terminal 12.

Under a first illustrated embodiment, a process will be described by which the remotely operated modeling system 10 may be used to construct and physically test electrical circuits. However, it should be understood that the described process is applicable to the assembly and/or control of any physical system, whether electrical, mechanical, or otherwise.

As a first step in understanding the system 10, a description will be provided of the control functions used by the system 10. Following a description of the control functions, a specific example will be provided of the use of the system 10.

The modeling system 10 includes a number of software and hardware components. Under the illustrated embodiment, a client program (e.g., a "client architecture" operating within the terminal 12) communicates with a server program (in the MRB 16), which in turn communicates with MRB resources 26, 28 that are used to perform a task. Data that is created during performance of the task is then returned to the client program in the terminal 12.

The client architecture operating within the terminal 12 consists of a graphical user interface, which the user employs to construct a graphical representation of a task to be performed. As the graphical representation is constructed, the program automatically builds the element list. When the user requests that the configured task be performed (i.e., executed), the element list is sent over a communication network (e.g. the Internet) to the server program (MRB).

After the elements list 20 is sent to the server 22, the server 22 assembles the elements of the list 20

and tests the modeled system (i.e., the "Device Under Test" or "DUT") as directed. The term "Device Under Test" or DUT is used herein to refer either to a single device or a collection of devices and physical elements connected as a physical system. Test results of the modeled system are collected by the server 22 and sent back to the terminal 12 as test data. The client program within the terminal 12 listens for incoming data from the server 22 and allows the data from the server 22 to be displayed in a variety of different ways.

As shown in Figure 7, when the Server program (Measurement Request Broker) first starts (i.e., begins running), it establishes a ServerSocket to listen for incoming Measurement Requests. A ClientSocket is established for communication with each separate Client that requests service.

In Figure 2, objects shown as rounded rectangles have a single instance (i.e., exist as a single copy), which is permanent. Objects shown as rectangles are transient and exist only during a single Measurement Request transaction.

The Client program sends to the Server program a text file (i.e., a Netlist shown in Fig 2) containing the element list 20. In the current embodiment the format of the text file uses the SPICE syntax. In another embodiment the element list may be constructed in the server as a result of separate messages sent from the client detailing modifications of the requested task. The MRBCompiler object 100 converts the Netlist into an MRBCircuit 102, which is forwarded to the MRBResourceAllocator 108. The MRBCircuit 102 contains a list of MRBDevices 112 corresponding to user-requested or user-desired devices.

The MRBResourceAllocator 108 converts this list of requested devices to an MRBMeasurement 106, which contains another list of MRBDevices 114, this time corresponding to devices that are actually available.

The MRBMeasurement 106 also includes a list of MRBActuators 116, a list of MRBResources 118 and a list of MRBSources 120. An MRBResource 118 encapsulates (i.e., provides the capabilities of) an actual device, in the form of an MRBDevice 114, but it can also provide information regarding the measurement capabilities supported by the device. Measurement capabilities include such things as "voltage sweep" or "measure current". The MRBActuators 116 allow a uniform access to devices 114 since they all support the same set of five basic operations: Clear, Setup, Wait (for external trigger), Commit (all changes), and Trigger. A device is wrapped in an MRBActuator (i.e., its operation is specified by the programming of the MRBActuator), which controls looping and operational flow of the device. The MRBMeasurement 106 knows how to use the MRBActuators (via programming limits resident within the MRBMeasurement), which in turn know how to control individual MRBDevices (also via a set of operational instructions/limitations). The separate list of MRBDevices in the MRBMeasurement correspond to passive devices within the list of MRBResources.

After the MRBMeasurement 106 has been created, the "connect()" method is called on this object, which causes the matrix switch 24 to be configured so that the desired connections are made. The Main program then calls the measure() method of the MRBMeasurement, which triggers the dominant MRBActuator. A text string (Data) is returned which contains the measured data. This Data string is returned to the Client program.

The client (i.e., terminal 12) may be built with Java
and provides facilities to manipulate the data from the measurement.

The steps of assembling and testing the DUT may be accomplished as shown schematically in FIGs. 2 and 7. The incoming data is received by the server 22 and a task object is spawned. The task object is used to encapsulate details of a physical task that will be performed under the direction of the task object. The task object is composed of user requested elements and is passed to a resource allocating object. Here the task (and its elements) are checked to make sure that all the parameters are acceptable, and the locations of the test equipment and elements of the DUT in the matrix switch are located. If all parameters are within acceptable ranges (i.e., within both instrument and element limitations) and the resources 26, 28 are available a modified task object is constructed. The matrix object is called to assemble the DUT, electrically connect any forcing function and the test equipment to the DUT and the modified task object is instructed to perform the measurement. The results of the measurement or error messages are returned to the server after the measurement has been performed. At this point, the data is serialized and the results sent back to the client within the terminal 12. Once the data has been received by the terminal 12, the user may manipulate it with the tools provided by the send client.

Under one illustrated embodiment of the modeling system 10, a user may connect to a web site 32 using a web browser installed within the terminal 12. One of the web pages to which the user may connect may contain

a Java applet that may be downloaded to the browser and run within the browser on the user's terminal 12. In another embodiment (discussed above), the client runs as a Java application, which does not require a web browser. The Java applet or application contains a user interface, which allows the user to graphically create a representation of a task to be performed. While many different applications may be used to create the representation (i.e., of the DUT), the circuit modeling language SPICE has been found to work well.

For example, the user may successively select (FIG. 3), elements from a palette (e.g., within a reference area 40), containing a collection of graphical symbols that represent building blocks of a task. In the case of an electronic circuit experiment, the different icons may represent electronic components (e.g., resistors, capacitors, inductors, transistors, etc.) and function blocks such as voltage sources (forcing functions) and measuring probes. The user may drag icons to an assembly area 42, one at a time (FIG. 4). The user interconnects these on the screen to represent the task to be performed (i.e., the modeled system, or DUT, to be constructed and tested). One or more parameters of the elements of the DUT can be provided by the user through the terminal 12.

Parameters are provided for a particular element by selecting the element with a mouse cursor and changing the default values of device parameters that appear in a dialog window 46 which appears. When the DUT (circuit 44) (or other task) has been configured, it may be saved to a file for later use. Similarly, a previously created file may be loaded from storage, or may be downloaded over the network allowing a person to quickly use a previously configured arrangement of components 50 (FIG.

5) as the DUT. The task is initiated by selecting the "measure" button 48 on the user's interface. Under the illustrated embodiment, a text-based description (i.e., an element list) of the task is sent over the network to the server 22. Further, the element list 52 (FIG. 6) may be examined at any time by activating a "netlist" button 54.

Once received, the server 22 parses the text-based description of the element list to check for syntactical correctness and then if the text can be parsed correctly, the server checks for the availability of required resources 26, 28 (i.e., resistors, capacitors, inductors, transistors, etc.). The client 18 may perform some of the checking for resource availability. The syntax used for the text-based description of the task may be that of the circuit simulation language SPICE. This has the added benefit that the task may be simulated within the terminal 12, or in some networked server, in advance of physical assembly and testing within the MRB 16.

Assuming that the resources are available, the server 22 locates the specific resources 26, 28 to perform the requested task; configures the required interconnections; configures the instruments; and triggers the instruments to collect data. After the measurements are completed, the data is read in from the instruments to the server. The server 22 repackages the data into a format suitable for the client and sends a data string back to the client programs. The client program parses the data string and internally stores that data in a format that can be presented to the user in both tabular and graphical form (FIG. 9).

In another embodiment of the system 10, distributed software object technology may be used to avoid the

creation and parsing of the text-based task description. An object stub in the client allows the direct creation of the type of task object that the server creates following the parsing step. (By distributed software object technology we means environments that follow the CORBA standard (ref: Object Management Group) or which use Microsoft's Distributed Component Object Model (DCOM).)

Turning now to the server 22, a description will be provided of the various threads followed in evaluating a DUT. The main server program thread first creates a listening socket and waits for incoming connections. When a connection is received the main thread creates a new socket representing this connection. The new socket is converted into a RWSocketPortal and various information items may be received from the client including the SPICE netlist. The main thread passes the netlist to the MRBCompiler where it is checked for syntactical and semantic correctness. The result of the compilation is a list of MRBCircuits. This list of MRBCircuits provides one embodiment of the task object mentioned earlier.

The list of MRBCircuits is passed to the MRBResourceAllocator by the main thread. In the MRBResource the MRBCircuits are checked to see if there are resources 26, 28 available to create the given circuit(s) (i.e., the DUT). The result of MRBResourceAllocator is that the MRBCircuits are converted into a set of MRBMeasurements. This set of MRBMeasurements is an embodiment of the modified task object mentioned earlier. The main thread then iterates through the list of MRBMeasurements calling each one's measure function and waits until the data is returned. After all the MRBMeasurements have been performed, the

resources allocated in the MRBMeasurements are returned to the MRBResourceAllocator. Then the main thread sends all the data collected back to the client. This process is repeated for the next client.

The netlist sent to the server 22 includes interconnect information necessary to assemble the DUT. As discussed above, the syntax used by SPICE indicates which device terminals are connected to which node in the circuit. Under the illustrated embodiment, circuit nodes are mapped into rows in the matrix switch 24.

For example, the following element list entries may be used to describe the physical system of the DUT shown in FIG. 8.

```
Q1    5     4     1     _CA3086Q1
Q2    2     0     1     _CA3086Q2
I1    1     0     DC    -0.1MA
R1    5     3     _10KOHM
R2    2     3     _10KOHM
V1    3     0     DC    10.0V
V2    4     0     SIN (0.0V 0.02V 1000.0Hz 0.0US 0.0
0.0DEG)
.TRAN 1US 10000.0US
.PRINT TRAN V(5) (-10.0V,10.0V)V(4) (-1.0,1.0V),
```

where Q1, Q2, R1 and R2 are structural elements of the DUT and I1, V1 and V2 are forcing functions of the DUT. This could result in the following resource assignments.

```
OSCOPEO->OSCOPE1     I1->CHAN2        Q2->id8
5, 29                1, 18            2, 36
0, 31                0, 16            0, 35
OSCOPE1->OSCOPE2     V2->SRS          1, 34
4, 30                4, 28            R1->id11
0, 31                0, 31            5, 41
V1->CHAN1            Q1->id7          3, 42
3, 17                5, 32            R2->id12
0, 16                4, 33            2, 43
                     1, 34            3, 44
```

In the resource assignment listing (shown above) each resource assignment (using the format "circuit label" -> "MRBResource") is followed by two or three matrix switch closure entries having the form "row number, column number".

For example, the DC source resource (forcing function) CHAN1 is used as an element in the circuit labeled V1. CHAN1 is connected to matrix switch columns 16 (-terminal) and 17 (+terminal). To provide the structural interconnections of the circuit labeled V1 (i.e., to connect the +terminal to node 3 and the -terminal to node 0 in the circuit), two matrix switches are closed. One connects column 17 to row 3 (shown by the entry 3, 17) and the second connects column 16 to row 0 (shown by the entry 0, 16). Similarly, the MRBResourceAllocator has allocated a transistor with internal label id7 for the circuit element labeled Q1. The three leads of the id7 transistor are connected to columns 32, 33 and 34. These three terminals are connected to nodes 5, 4 and 1 in the circuit by closing relays, which connect columns 32, 33 and 34 to rows 5, 4 and 1, respectively.

The matrix switch 24 is a two-wire switch. The external connections to the switch 24 are shielded coaxial connections. The shield is connected to one of the wires and the center conductor to the second. The two conductors become joined at the device terminal. At the instrument terminals, the two wires are handled differently depending on whether the instrument is making a DC or a transient measurement. For transient measurements, the shield is left floating, whereas for DC instrument measuring points, the two wires are connected to the force and sense terminals on the instrument. This enables a high accuracy "four-wire" Kelvin measurement to be made which corrects for resistance losses in the leads. A more detailed description of the individual classes and methods used in the server program is provided in the attached Appendix I.

The modeling system 10 is a public facility on the Internet and vulnerable to attacks and malicious use. Two major precautions have been put in place to safeguard the modeling system 10. The first precaution is that the user is never aware of which TCP port the system 10 is monitoring for measurement requests. This is done to prevent anyone from directly accessing the measurement server 22 without using an official client (downloaded from the server 22).

The second precaution lies in the MRBResourceAllocator of the server 22. Here the task is checked to make sure that: all parameters in the measurement task do not exceed the tolerances specified for the instruments and DUT in the SQL database or internal program database. With this validation by the MRBResourceAllocator, no unsafe measurements are allowed

to be performed. This process functions to protect the measurement equipment and DUTs from misuse.

Since the modeling system 10 is available to anyone using a web browser on the Internet, the client interface must be intuitive and easy to use. The ease of use is achieved by providing convenient graphical methods for configuring different types of measurement. The client architecture allows additional graphical tools to be plugged in to the client. Thus a specific user interface can be designed for a given measurement type, and used with a generic or specifically designed data presentation module. This feature also allows the modeling system 10 to be extended to perform highly specialized measurements.

For example, the modeling system 10 may be extended to a manufacturing operation where the DUT is a model, which has a predetermined form representing the manufacturing operation. The model may be downloaded to the terminal 12. A user may alter the model and observe the effects in real time.

A specific embodiment of a method and apparatus for a method modeling physical systems according to the present invention has been described for the purpose of illustrating the manner in which the invention is made and used. It should be understood that the implementation of other variations and modifications of the invention and its various aspects will be apparent to one skilled in the art, and that the invention is not limited by the specific embodiments described. Therefore, it is contemplated to cover the present invention and any and all modifications, variations, or equivalents that fall within the true spirit and scope of the basic underlying principles disclosed and claimed herein.